IN THE SPECIFICATION:

Please replace paragraph [0033] with the following amended paragraph:

[0033] Storage device 89 is a DASD (Direct Access Storage Device), although it could be any other storage such as floppy disc drives or optical storage. Although storage 89 is shown as a single unit, it could be any combination of fixed and/or removable storage devices, such as fixed disc drives, floppy disc drives, tape drives, removable memory cards, or optical storage. Main memory 87 and storage device 89 could be part of one virtual address space spanning multiple primary and secondary storage devices.

Please replace paragraph [0036] with the following amended paragraph:

[0036] In one embodiment, rather than construct a wide ALU (e.g. 64 bits) with one monolithic block of combinational logic, a wide ALU is split into an upper ALU and a lower ALU ~~107~~ (e.g., 32 bits each), with upper and lower portions of the operands input to their respective ALU. The lower ALU produces a carry out signal that indicates the result from the upper ALU needs to be incremented. When this carry out is false, no additional action is take. When this carry out is true, it is used to stall the pipeline in order to introduce a clock cycle to perform the increment. This is particularly effective for adding offsets to base addresses. In one aspect, a carry out signal is generated from a bit location near a middle bit position within a wide ALU, and this carry out signal is used to stall at least part (i.e., some stages) of the pipelined logic.

Please replace paragraph [0037] with the following amended paragraph:

[0037] In another embodiment, a fast forwarding bus carries data that assumes the carry out is zero (i.e., no carry) and that the upper ALU can be bypassed. The fast forwarding bus is useful for cases where the full result is needed on the next clock cycle. The fast forwarding bus holds an "only speculatively correct" result for the wide ALU, and the determination as to whether this result can be used on the next clock cycle is determined by the carry out signal. This "speculatively correct" result can be written to various memories, and then only used on subsequent clock cycles if determined to be correct. This embodiment takes advantage of recognizing when the most significant bits of either operand are all zeros. Since the upper ALU is bypassed by one of the operands, that part of the operand can be forwardi~~nge~~d to: (1) the next unit or ALU; and (2) ~~feedback~~ fed back to upper register bits; and furthermore, the bypassing allows that part of the operand to reach its destination well within one clock cycle. For this reason, very small and area efficient wires can be used for these bits on the fast forwarding bus, thereby saving power and die area.

Please replace paragraph [0043] with the following amended paragraph:

[0043] A more detailed embodiment of the ALU 200 is shown in Figure 3. The figure shows the input for the upper ALU 105 and lower ALU 107 coming from registers 201, 203, 205, 207. These registers 201, 203, 205, 207 receive input from busses 211, 213, which are driven from a plurality of sources. This particular embodiment shows the busses 221, 223 driven by multiplexers 211, 213, 215, as well as other sources. The lower ALU 107 outputs its result onto two busses 231, ~~333~~ 233: (1) the bus 231 used to write to the output register 135; and (2) the fast forwarding bus 233. The upper ALU 105 two buses: (1) the bus 235 used to write to the output register 135; and (2) the fast forwarding bus 233. Depending on implementation details the three busses 231, 233, 235 could be a single bus. An instruction register 237 for controlling the operation of the ALUs 105, 107 is also shown in the Figure 3.

Please replace paragraph [0047] with the following amended paragraph:

[0047] Since the upper bits (from either RAH 201 or RBH 205) do not go through the ALU, they have ample time within a clock cycle to be forwardi~~nge~~d to: (1) the next unit or ALU; and (2) ~~feedback~~ fed back to upper register bits (RAH 201 and/or RBH 205). For this reason, very small and area efficient wires can be used for these bits on the fast forwarding bus 235, thereby saving power and die area.

Please replace paragraph [0049] with the following amended paragraph:

[0049] As an aid to use the fast forwarding bus 235, it would be advantageous to know whether RAH 201 and/or RBH 205 contain all zeros at the beginning of the clock cycle, thereby aiding in limiting the stall for a forward to one cycle. As exemplified in the register file 300 shown in Figure 4, this is achieved by keeping a flag bit Z 301 in each memory word 303 (similarly included in other memories, such as registers, etc.) that indicates the high order bits are all zeroes. Arithmetic units can be designed to generate this condition code 301 along with the usual data and condition codes. If, when an addition is to be performed, the flag bits 305, 307 for the two operands are examined, and the upper adder 105 is not used if either flag 305, 307 indicates the presence of all zeros. The fast forwarding generates the correct result when there is not a one value at carry out 141. To further limit potential unexpected carry out 141 one values from the lower adder 107, the zero flag can be extended to indicate that more bits of the value is all zeros. For example, in a 64-bit ALU, the Z bit 301 could indicate that the upper 40 bits are all zeros, reducing the rate of unexpected carry outs by a factor of 2^8= 256.